Host Identity Protocol (HIP): Connectivity, Mobility, Multi-Homing, Security, and Privacy over IPv4 and IPv6 Networks

Pekka Nikander, Andrei Gurtov, and Thomas R. Henderson

***Abstract*—The Host Identity Protocol (HIP) is an inter-networking architecture and an associated set of protocols, developed at the IETF since 1999 and reaching their first stable version in 2007. HIP enhances the original Internet architecture by adding a name space used between the IP layer and the transport protocols. This new name space consists of cryptographic identifiers, thereby implementing the so-called identifier / locator split. In the new architecture, the new identifiers are used in naming application level end-points (sockets), replacing the prior identification role of IP addresses in applications, sockets, TCP connections, and UDP-based send and receive system calls. IPv4 and IPv6 addresses are still used, but only as names for topological locations in the network. HIP can be deployed such that no changes are needed in applications or routers. Almost all pre-compiled legacy applications continue to work, without modifications, for communicating with both HIP-enabled and non-HIP-enabled peer hosts.**

The architectural enhancement implemented by HIP has profound consequences. A number of the previously hard networking problems become suddenly much easier. Mobility, multi-homing, and baseline end-to-end security integrate neatly into the new architecture. The use of cryptographic identifiers allows enhanced accountability, thereby providing a base for easier build up of trust. With privacy enhancements, HIP allows good location anonymity, assuring strong identity only towards relevant trusted parties. Finally, the HIP protocols have been carefully designed to take middle boxes into account, providing for overlay networks and enterprise deployment concerns.

This article provides an in-depth look at HIP, discussing its architecture, design, benefits, potential drawbacks, and ongoing work.

***Index Terms*—Inter-networking, Mobile communication, Multi-access communication, Communication system security, Communication architecture, Host Identity Protocol, Identifier/Locator split**

I. Introduction

THE HOST Identity Protocol (HIP) and architecture [1], [2] is a new piece of technology that may have a profound impact on how the Internet will evolve over the coming years. The original ideas were formed through discussions at a number Internet Engineering Task Force (IETF) meetings during 1998 and 1999. Since then, HIP has been developed by a group of people from Ericsson, Boeing, HIIT, and other companies and academic institutions, first as an informal activity close to the IETF and later within the IETF HIP working group (WG) and the HIP research group (RG) of the Internet Research Task Force (IRTF), the research arm of the IETF.

From a functional point of view, HIP integrates IP-layer mobility, multi-homing and multi-access, security, NAT traversal, and IPv4/v6 interoperability in a novel way. The result is architecturally cleaner than trying to implement these functions separately, using technologies such as Mobile IP [3] [4], IPsec [5], ICE [6], and Teredo [7]. In a way, HIP can be seen as restoring the now-lost end-to-end connectivity across various IP links and technologies, this time in a way that is secure and supports mobility and multi-homing. As an additional bonus, HIP provides new tools and functions for future network needs, including the ability to securely identify previously unknown hosts and the ability to securely delegate signalling rights between hosts and from hosts to other nodes.

From a technical point of view, the basic idea of HIP is to add a new name space to the TCP/IP stack. These names are used above the IP layer (IPv4 and IPv6), in the transport layer (TCP, UDP, SCTP, etc) and above. In this new name space, hosts (i.e., computers) are identified with new identifiers, *Host Identifiers*. The Host Identifiers (HI) are public cryptographic keys, allowing hosts to authenticate their peer hosts directly by their HI.

HIP can be considered as one particular way of implementing the so-called identifier / locator split approach [8] in the stack. That is, while in the current IP architecture the IP addresses assume the dual role of acting both as *host identifiers* and *locators* (see Section IV-B), in HIP these two roles are cleanly separated. The Host Identifiers take, naturally, the host identifying role of the IP addresses while the addresses themselves preserve their locator role.

As a result of adding this new name space to the stack, when applications open connections and send packets, they no longer refer to IP addresses but to these public keys, i.e., Host Identifiers. Additionally, HIP has been designed in such a way that it is fully backwards compatible to applications[1] and the deployed IP infrastructure. Hence, for example, when an existing, unmodified e-mail client opens a connection to the e-mail server hosting the mailbox, the e-mail client hands over a

Manuscript received 24 September 2009; revised 5 April 2009.

P. Nikander is with NomadicLab, Ericsson, Jorvas FIN-02420, Finland (e-mail: pekka.nikander@nomadiclab.com).

A. Gurtov is with Helsinki Institute for Information Technology, Helsinki University of Technology, P.O. Box 9800, FIN-02015 TKK, Finland (e-mail: gurtov@hiit.fi).

T. Henderson is with the Boeing Company, P.O. Box 3707, Seattle, WA, USA (e-mail: thomas.r.henderson@boeing.com).

Digital Object Identifier 10.1109/SURV.2010.021110.00070

[1]The application level backwards compatibility does not necessarily apply to diagnostic and other system administration related applications, depending on how they use IP addresses.

reference to the public key[2] to the operating system, denoting that it wants the operating system to open a secure connection to the host that holds the corresponding private key, i.e., the e-mail server. The resulting connection can be kept open even if both of the hosts, i.e., the client and the server, are mobile and keep changing their location. If the hosts have multiple access links in their disposal, HIP allows these multiple links to be used for load balancing or as backups, invisible from the applications.

To deploy HIP in a limited environment, all that is required is to update the involved hosts to support HIP. No changes are required to typical applications or to the IP routing infrastructure; all nodes will remain backwards compatible with existing systems and can continue to communicate with non-HIP hosts. For full HIP support, it is desirable to add HIP related information to the Domain Name System (DNS). In addition, a new infrastructure service is needed to support HIP rendezvous services (see Section V-B). If it is impossible to upgrade the operating system of some particular host, e.g., a legacy mainframe, it is also possible to add a front-end processor to such a system. The front-end processor acts as a HIP proxy, making the legacy host to appear as (a set of) HIP host(s) to the rest of the network.

HIP is currently published as an experimental IETF standard, with the publication of the RFCs describing the protocols, main extensions, and infrastructure support in early 2008. The architecture itself is described in RFC 4423 [1], published in 2006. There are three independent open-source implementations of HIP, and an active, growing user community.

In this article, we describe the history behind HIP, the HIP architecture, the associated protocols, the potential benefits and drawbacks for prospective users, and ongoing work. Although existing surveys [9], [10] briefly described HIP in comparison with other mobility and multihoming protocol, no comprehensive tutorial paper had yet been published, to our knowledge.

The rest of the article is organised as follows. First, in Section II, we briefly describe how the environment has changed since the Internet architecture and protocols were originally developed. Consequently, in Section III, we discuss some major problems in the current Internet: loss of connectivity, poor support for mobility and multi-homing including problems with multicast, surge of unwanted traffic, and lack of authentication, privacy, and accountability. These problems are mainly a result of the changing environment and requirements; HIP changes the architecture, helping to solve the problems. In Section IV we describe the HIP architecture and related protocols. In Section V, we discuss how HIP helps with connectivity, mobility, and multi-homing over IPv4 and IPv6 networks. Next, in Section VI we briefly look back at unwanted traffic and lack of authentication, privacy, and accountability, and discuss how HIP could help in solving those. In Section VII, we discuss the maturity status of HIP, reporting where it is currently being used, the standardisation situation, and existing implementations. Section VIII concludes the paper.

[2]How exactly an application refers to the Host Identity public keys depends on implementation. In most implementations today, the applications use Host Identity Tags (see Section IV-C), which look like IPv6 addresses.

II. Background: Evolving Environments

The original design of the TCP/IP Internet protocols was created for an environment where the end-users were assumed to be mutually trusting, at least to a minimal degree, and where the network is assumed to be inherently unreliable due to a potential attacker physically destroying routers and links [11]. Since then, the environment has grossly changed as a side effect of the huge success of the Internet, creating a need to devise a communication architecture that provides the following functions:

- Ability to operate over all kinds of underlying networks, including ad hoc, commercial, and dedicated; this implies the ability to dynamically pay for the services on-line, the ability to hide the real identities of communicating parties from the underlying networks, etc.
- Ability to survive in a partially hostile environment where some of the underlying networks may be only partially co-operating, competing, or even outright antagonistic to each other; this implies the ability to isolate underlying networks from each other, when needed.
- Ability to support application, host, and sub-network level mobility and multi-access as primary design elements and not as extensions.
- Ability to support full location privacy, especially against any transit networks and other third parties.

The goals above can be seen as a new incarnation of the original IP design goal, adapted to the contemporary needs. Nowadays the underlying communication network is more diverse, sometimes even hostile, in addition to being unreliable, and a fraction of users must be assumed to be egregiously selfish or outright malicious.

Along with revising the original goals to meet today's needs, it has also become clear that the operational costs of the current network are becoming quite high. Consequently, there is a need for a network that can self-organise, including functions such as infrastructure discovery and the ability to find reasonably functioning communication paths among multiple alternatives.

The HIP architecture and base protocols aim towards the aforementioned goals. While they do not, as such, provide all functionality that is needed to fulfill the goals, they create a new inter-connectivity layer that spans over both IPv4 and IPv6, relying heavily on the availability of IP infrastructure but, in theory at least, being also able to run over non-IP links and media. They provide baseline protection for communication, including optional functionality to fully protect the identity of communicating parties from outsiders. They contain a set of basic mechanisms to support host mobility and multi-homing, allowing these mechanisms to be adopted and extended to better fit to differing environments. Finally, they aim to provide a similar or higher level of flexibility than the original IP architecture did, allowing the protocols and mechanisms to be easily extended.

III. Fundamental Problems

Before diving to the details of the HIP architecture and protocols, we take a brief look at a few of the most challenging problems in the contemporary Internet: loss of universal connectivity, poor support for mobility and multi-homing, unwanted traffic, and lack authentication, privacy, and accountability. This forms a baseline for Sections V and VI, where we explain how HIP and other mechanisms relying on HIP alleviate these problems. We do not claim that HIP is the only solution for these problems; many different protocols such as Mobile IP, MobIKE, TLS/DTLS, DNSSEC, etc. have been defined to solve some subset below. We believe, however, that HIP offers an interesting and unique approach to providing an integrated solution to all of the problems listed below.

At the same time, it must be understood that HIP does not provide much remedy, at least not currently, to a number of other hard problems in the Internet, such as self-organising networks and infrastructure, or intermittent connectivity. On the other hand, there may certainly be new, still-unfound ways to utilise HIP to make those or other hard problems easier to solve.

A. Loss of universal connectivity

Compared to the original Internet, perhaps the largest problem all current ones is the loss of connectivity, caused by NATs, firewalls, and dynamic IP addresses. This issue is so familiar to everyone using the Internet that it may not appear a problem at all; almost no-one any more expects to be able to connect to another host at will. The so-called Internet transparency or the end-to-end problem refers to the fact that the Internet no longer has a universal addressing scheme as it originally did [12].

The so-called classical network-layer addressing invariants (see Section IV-A) have been eroded by the use of private address space (NAT for IPv4) and by the introduction of IPv6 which is incompatible with IPv4 on the wire [1]. For the foreseeable future, the networking landscape will be dominated by a mixture of addressing realms, including private and public IPv4 and IPv6 address spaces, and there will be a desire for hosts in one realm to access services in another. Attempts to deal with these realms at the network layer via NAT [13] and protocol conversion boxes (NAT-PT) [14] have proved to be fragile, raft with security issues, and architecturally undesirable (see, e.g., [15]). The HIP architecture offers a possible path for overcoming these limitations by providing a new end-to-end naming invariant and protocol mechanisms that allow the IP addresses used on a wire to be used as ephemeral locators rather than host identifiers themselves.

The Internet architecture has historically leaned away from providing explicit support for policy-enforcing middle-boxes (firewalls) that control network-level access between network segments. Nevertheless, operational networks today are reliant on firewalls that attempt to enforce network policies to allow or deny traffic based on IP addresses and transport port numbers. The security robustness of current approaches is challenged by the fact that it is difficult for such firewalls to control what is actually sent on a given port, and to attribute a particular IP address to a particular legitimate host rather than an impostor or inside attacker who acquired an address. Usually, there is little or no communication between the end hosts and the firewalls, leading to fragile escalation techniques whereby end hosts try various tricks to get through such restrictions.

Many aspects of the HIP architecture, together with a number of design details related to middle boxes, e.g., the ability of hosts to directly authenticate themselves to firewalls via explicit registration or implicit overhearing of a control handshake (see Section V-D), are results from trying to re-establish, in a controlled manner, the original transparency principles of the Internet. This aim, together with the requirements caused by the changed environment, can be seen as the main driving forces behind the design decisions of HIP.

B. Poor support for mobility and multi-homing

Effective mobility support requires a level of indirection[3] [8], to map the mobile entitys stable name to its dynamic, changing location. Effective multi-homing support (or support for multi-access / multi-presence) requires a similar kind of indirection, allowing the unique name of a multi-accessible entity to be mapped to the multitude of locations where it is reachable.

Within the Internet community, the historical approach to solve these problems has been to consider mobility and multi-homing as separate, technical problems, something that just needs to be solved through engineering. The main result of this attitude are the Mobile IP protocols [3], [4], which are architecturally based on re-using a single name space, the IP address space, for both stable host identifiers (Home Addresses) and dynamic locators (Care-of Addresses). While the approach works in basic network topologies, it creates two major drawbacks. Firstly, it binds the communication sessions (TCP connections and application state) to the home addresses. This, in turn, when combined with the only known scalable solutions to a number of related security problems, creates an undesirable dependency on a constant reachability of the home address. In other words, the mobile host is intrinsically bound to the availability of the home addresses; the home agent becomes a new single point of failure[4].

Secondly, approaches that use names from a single name space for multiple purposes create a number of potential semantic problems [16]. For example, the so-called alias problem [17] relates to the use of multiple names from a single name space to denote same entities in a non-transparent way. In practical terms, when Mobile IP is used, there is no easy way to tell if two IP addresses point to a single host (e.g., due to one being its home address and another one its care-of address) or not, i.e., whether one is merely an alias for the other or an identifier for a genuinely different host. For applications or users that cache previously used IP addresses and reuse them later, aliasing can cause applications to unknowingly connect to different hosts. On the other hand,

[3]The existing layer of indirection, i.e., mapping from DNS names to IP addresses, is insufficient for two reasons. Firstly, not all applications use the DNS, and even those that use it often cache or store IP addresses internally. Secondly, in real life updates to DNS data may take more than 30 minutes to get globally distributed, making it too slow to provide a solution for many realistic mobility situations.

[4]From a practical point of view, the Mobile IP Home Agent being a single point of failure may not be really a problem. It is a matter of system engineering to overcome the problem through distributing the functionality over a number of distinct nodes. However, to distribute the Home Agent functionality over a number of sites, help from the routing system is needed, thereby placing some extra burden upon it, re-creating the original mobility and multi-homing problem in a recursive manner.

multi-homing creates the inverse problem, where aliasing (multiple IP addresses pointing to a single host) is the desired outcome but the applications are not aware of it.

As briefly mentioned above, HIP provides an alternative approach to implementing mobility and multi-homing. It explicitly adds a new layer of indirection and a new name space, thereby adding the needed level of indirection to the architecture. Furthermore, the inherent ability to delegate, provided by the cryptographic nature of the Host Identifiers, allows HIP to provide more natural support for other granularities of mobility, such as application or sub-network mobility (see Sections V-E and V-F).

C. *Problems with multicast*

Certain applications, such as Internet TV, involve data transmission from one source to multiple destinations. Other applications, such as multiparty video conferencing, involve transmission from several sources to several destinations. Such scenarios are most efficiently handled using multicast data transmission, where the source transmits a single copy of data and routers or hosts in the network multiply packets as needed for delivery to downstream recipients.

The multicast can be implemented on the networking layer as native IP multicast, or as an application service on the overlay network. The network multicast is more efficient than the application multicast, because it can achieve ”one link – one packet” principle, whereas application multicast can still transmit multiple copies of the same packet of a link. Application multicast is easily deployable while several issues hindered deployment of IP multicast.

Mobility of hosts participating to multicast is a largely unsolved problem. Especially, if the multicast source changes the IP address, the whole multicast tree needs to be reconstructed. Two common approaches for multicast receiver mobility are bidirectional tunnelling, based on Mobile IP [3], which tunnels multicast data to and from the home and visited networks, and remote subscription [18], in which the visiting multicast receiver joins the local multicast tree. Some solutions to provide authentication to multicast receivers have been proposed. However, they are only able to authenticate subnetworks where the user is located, but not the host itself. Therefore, other hosts from the same subnetwork can receive the stream without authentication. Researchers have started to explore whether HIP can help also to alleviate these problems by allowing hosts to be more directly authenticated by the network [19].

D. *Unwanted traffic*

The various forms of unwanted traffic, including spam, distributed denial of service (DDoS), and phishing[5], are arguably the most annoying problems in the current Internet. Most of us receive our daily dosage of spam messages; the more lucky of us just a few of them, the more unlucky ones a few hundreds each day. Distributed denial of service attacks are an everyday problem to large ISPs, with each major web site or content provider getting their share. Phishing is getting increasingly common and cunningly sophisticated.

The current Internet architecture can be considered as a distributed extension of the well-known message passing interprocess communication paradigm [20]. That is, within a single centralised computer, an operating system typically provides resources for and isolates a number of concurrent, parallel processes, each running a separate program or performing a specific task. In any modern operating system, the communication between the processes is strictly controlled by the operating system, in order to provide a level of sanity, thereby preventing bugs or malicious code in one program to cause (much) harm to other programs.

In the message passing paradigm, a sending process creates a message and sends it to a specific other process, by naming the recipient with a name. The current packet-based networking, as we know it from the Internet, can clearly be considered as an extension of this message passing Inter-Process Communication (IPC) paradigm. In both the paradigm and in the Internet, there is a sender (whether a process or a host) that creates a message, names a recipient, and asks for the underlying mechanisms to pass the message to the recipient. However, what is noteworthy here is that current Internet implementation gives all power to the sender. Initially, when the sender creates a message and dispatches it, the network has no idea of whether the recipient will be interested in the message. Only when the message arrives at the named receiving host (or its proxy, such as a firewall), the recipients consent is consulted. Only then are unwanted messages dropped.

Basing on this and looking at the situation from an economic point of view, we can characterise the current Internet as a global, distributed message passing IPC system where the main cost of unwanted communication is paid by the recipient. This is a direct (though certainly unintentional) consequence of the network architecture. By explicitly and directly naming all potential recipients, we create a system where the senders can easily act on their desire to send data to any recipient in the network. Given that under the typical flat-fee contracts the marginal cost of sending additional packets is very close to zero (up to some capacity limit), there are few or no incentives for refraining from sending unwanted traffic; sending some more packets, either just for fun to gain legitimate or illegitimate profits, costs so little that it does not matter. Hence, for SPAM, even a marginal response rate creates a strong incentive for sending unsolicited advertisements, and for DDoS-based extortion, even a small success rate creates a strong incentive to launch attacks [21].

To summarise, our claim is that the current unwanted traffic problem is a compound result from the following factors:

- An architectural approach where each recipient has an explicit name and where each potential sender can send packets to any recipient without the recipient's consent.
- A business structure where the marginal cost of sending some more packets (up to some usually quite high limit) is very close to zero.
- The lack of laws, international treaties, and especially

[5]The problem of phishing is more complex than the other two main harmful forms of unwanted traffic, DDoS and spam, for reasons that go beyond the scope of this paper. The main economic argument related to unwanted traffic relates also to it, even if it leaves many other aspects aside. The related aspects of authentication and attribution are discussed in Section III-E

enforcement structures that would allow effective punishment of those engaging in illegal activity in the Internet.
- The basic profit-seeking human nature, driving some people to unethical behaviour in the hopes for easy profits.

Of course, there is nothing that one can do with the last co-factor (human nature), other than to accept it. The third one is more regulatory in nature and therefore falls beyond the scope of this article. For the other two, the separation of identifiers and locators can be used to create architectures where a sender must acquire the recipients consent before it can send data beyond severely rate-limited signalling messages (see Section VI-B).

E. Lack of authentication, privacy and accountability

The aim of authentication, privacy, and accountability is to prevent organisationally or socially undesirable things from happening, on one hand by imposing technical restrictions on information flow, and on the other hand by creating explicit incentives for desirable behaviour. Hence, in some sense our aim here is similar in nature to our goal with reducing unwanted traffic.

Today, the authentication problem is technically easy but socially very challenging to solve. Except for semi-widespread use of TLS certificates for HTTP, there are no widespread, interoperable authentication techniques in use in the Internet, even if S/MIME [22], IPsec [5], etc., have been available and widely implemented for a decade or longer. The main problems related to the difficulty of creating trust between users and organisations.

The privacy problem is a complex one, with at least three different viewpoints. From the Orwellian point of view, the question is about freedom of speech and governmental control. A sufficient privacy system ensures that we can express our opinions and think freely, within reasonable bounds (like not committing clearly criminal acts) even when our opinions are socially unacceptable or hostile towards the governing regime. The Kafkaesque aspect of privacy focuses on citizens ability to retain their autonomy without fear of unfounded litigation or other harassing legal or other action [23]. Thirdly, the economic aspect of privacy relates to the fine balance between socially beneficial differentiated pricing versus socially harmful price discrimination[6] [24]. From these three different points of view, it seems a necessity to provide a reasonable base-level of privacy as a built-in feature in future networks.

The flip side of privacy is accountability. Unbounded privacy encourages irresponsible behaviour patterns, such as rampant advertising. To counter these, increased privacy requires increased accountability; a fact that appears as a paradox from the technical point of view. A key to understanding this technical paradox is to consider the different dimensions of communication. At the baseline level, we can make a difference between four dimensions: the *content* of communication, the *parties* communicating, their *locations*, and finally the very fact that a piece of communication took place (*existence*). If the system is able to provide strong insulation between these dimensions so that each party gets only the relevant pieces of information, a high level of privacy can be preserved. For example, anyone whose resources are used to carry packets needs to know whom to attribute the packets to, but should have no access the information content, the identity of the other parties[7], nor their locations.

At the same time, if these components can be combined, post hoc, as in the face of criminal activity, it becomes possible to provide accountability. For example, if a first party, seeing the content, does not know the real-world identity of the peer host and definitely not its location, if a second party, seeing the identities, does not know the contents of the communication nor the locations of the communicating parties, and a third party, knowing the locations, has no clue about contents nor identities, the system can be engineered to be highly privacy protecting. If, at the same time, it is possible to combine the knowledge from these three parties, e.g., through manual actions or relatively pricey cryptographic operations, accountability becomes possible in a way that makes privacy violations hard and costly.

While the Host Identity Protocol does not directly provide means to address the privacy and accountability problems, it changes the landscape in a number of ways. Firstly, the use of cryptographic host identifiers as an integral part of connectivity, thereby providing automatic identity authentication, makes it easier to attribute a series of acts to a distinct host. Secondly, the separation of identities and locators makes it easier to hide the topological location of communicating parties. Thirdly, there are a few privacy extensions to HIP [25] [26] that allow the identities of the communicating parties to be hidden from third parties.

IV. The HIP architecture and base exchange

With the changing networking environment, and a few problems for which the current architecture is not well-suited to solve, briefly described above, we now turn our attention back to the Host Identity Protocol itself. In this Section, we describe what HIP is in detail; after that, in the two next sections, we turn our attention on how HIP can be used as a tool in alleviating the above discussed architectural problems.

The original ideas for HIP architecture grew from the desire to provide a means for providing better support for security and mobility within the IP architecture. At the same time, from the beginning, there were alluring sensations that HIP might turn out to be much more, i.e., it might provide a means to heal the damage caused by NATs by providing a new name space for interconnectivity across distinct networks (see Sections III-A, V-C and V-D). Architecturally, HIP promises to fulfill those inklings and more. However, only time will

[6]There are widely differing views on when price differentiation is socially useful and when harmful; the issue often depends on the exact details of the markets. However, the whole issue falls beyond the scope of this paper. The interested reader is advised to the cited and other papers by Odlyzko.

[7]In an architecture that is both privacy protecting and where all traffic can be attributed to some real world entity, each access provider needs to know the identity of the host sending or receiving packets, each transit provider must be able to attribute the packets to the access providers, etc. However, there is no ex ante need for any single party to be able to attribute *both* the sender *and* the receiver of any single packet, or a packet flow, to respective real world entities. In contrary, such a practise creates an economic information asymmetry that more easily leads to socially negative, unconsented forms of traffic or price differentiation.

show if it has enough of appeal to make any larger effect on the networking reality of tomorrow.

The HIP architecture was carefully crafted to meet the known current requirements and simultaneously provide for enough flexibility for future adaptations. Over the years, the aim has broadened from a name space that works as a security and mobility tool, through generalising the mobility support to cover also multi-homing, towards a general sublayer that provides interconnectivity in the same way the original IP did. In other words, the current HIP aim can be characterised as providing the lowest layer in the stack that encompasses location-independent identifiers and end-to-end connectivity. Furthermore, HIP cleanly separates host-to-host signalling and data traffic into separate planes, i.e., it can act both as an interconnectivity-level signalling protocol and as a general carrier for higher-layer signalling. Thereby it has appeal for architectures where control is separated from data, e.g., due to commercial requirements.

Starting from the architectural ideas, i.e., the desire to add a new, secure name space and a new layer of indirection, HIP has been carefully engineered towards two goals that are in partial conflict with the architectural ideas. Firstly, it is inherently designed to utilise, in an unmodified form, the current IP-based routing infrastructure (IPv4 and IPv6). At the same time, it was engineered to support the current application networking APIs with sufficient semantic compatibility[8]. Hence, HIP is backwards compatible and deployable in parallel to the existing stack, requiring no changes to the routing infrastructure or to the typical user-level applications. Secondly, the goal has been to implement the desired new (or renewed) functionality with a minimal set of changes to the existing system. In practical terms, a HIP implementation that is integrated with an existing kernel-level TCP/IP typically requires only a few hundred lines of code modifications to the kernel[9]; all the rest runs in user space. Even the user level components are quite small, at least if compared to alternative ways of providing the same functionality (see Section IV-E for more details).

A. Approach

Architecturally speaking, the current division of the functionality into the IP and TCP/UDP layers appears to be suboptimal from security, mobility, and multi-homing points of view. For example, from a functional point of view, the upper parts of IP (up from and including IPsec) apparently belong more tightly to transport than the lower part of IP, i.e., the routing and forwarding part. In other words, the current division of work between the IP and transport layers seems to make a number of network functions, including mobility and multi-homing support, harder than necessary. At a more fundamental level, one can question whether the very division of work between two distinct layers, i.e., IP and transport, makes sense at all. For example, a prominent network architecture veteran, John Day, argues that the IP and transport layers form a single unit that should be considered as a single layer consisting of a number of sublayers [20].

HIP attempts to provide a partial fix for the layering problems by rearranging some of the functionality within the very core of TCP/IP. While doing so, it attempts to restore, in an enhanced form, the four classical network-layer addressing invariants, or the original characteristics of the IP addresses, when viewed as identifiers [1]:

- **Non-mutability:** The source and destination identities sent are the identities received.
- **Location independence:** The identities do not change during the course of an "association".
- **Reversibility:** A return header can always be formed by reversing the source and destination identities.
- **Omnisciency:** Each host knows what identities a peer host can use to send packets to it.

In the current world, we have been forced to give up all but reversibility; furthermore, we surmise that the only reason reversibility has been preserved is that the Internet would stop working without it.

Consider now Domain Name System (DNS) names. From the practical, functional point of view, DNS names are references to IP addresses[10]. As the majority of current applications are based on some variant of the socket API, the applications themselves (directly or within a library) resolve the DNS names to the corresponding IP address (and other information), and use the IP address in the socket API, to identify the destination host (and application).

The Host Identity (HI) name space, introduced by HIP (see Section IV-B, below), can be considered to fill an important gap between the IP and DNS name spaces. By creating a new inter-networking facility on top of the existing IP networks, HIP restores the classic invariants within the identity name space while freeing the routing system from the burden of even minimally preserving the original IP addressing semantics. That is, HIP allows the underlying communication layers, i.e., IPv4 and IPv6, to give up all but the third network-layer invariant. Furthermore, if a global, Host Identity based rendezvous service (see Section V-B) is added to the architecture, even the third invariant can be dropped.

As a result, if HIP were universally used by all Internet hosts, the IP addresses could become fully agile, i.e., they could be changed whenever needed while minimally disrupting existing or new communication associations, they could be fully determined by the location, there would be no need to use reversed addresses when sending traffic back to the originator, and there would be no need for a host application to know what addresses other hosts need to use to reach it.

B. Basics

As mentioned above, the core of HIP lies in implementing the so-called identifier / locator split in a particular way. As

[8]Economics is a main reason why the HIP authors have considered API backwards compatibility so important. Any change that requires all the applications to be changed (such as IPv6) is likely to take a very long time, due to the high cost of updating old, legacy applications. HIP aims towards a less painful path here, allowing most applications to continue to be used unmodified. New applications, of course, could utilise new APIs.

[9]The required modifications are included in recent versions of the vanilla Linux (2.6.21 or newer) and FreeBSD (7.X series) kernels.

[10]We are well aware of other uses of the DNS, such as SRV records [27]. However, SRV records are commonly used only with a limited number of protocols, such as XMPP, SIP, and LDAP. Furthermore, even then the name is eventually resolved to an IP address.

already briefly discussed, in traditional IP networks each host has an IP address that serves for two different purposes: it acts both as a *locator*, describing the current topological location of the host in the network graph, and as a host *identifier*, describing the identity of the host, as seen by the upper layer protocols [8]. Today, it impossible to use the same IP address for both purposes, due to the host mobility and multi-homing requirements.

A solution to this problem is to separate the identity and location information from each other. HIP separates the locator and identifier roles of IP addresses by introducing a new name space, the *Host Identity* (HI) name space. In HIP, a Host Identity is a public cryptographic key from a public-private key-pair. A host possessing the corresponding private key can prove the ownership of the public key, i.e., its identity. As discussed in more detail in Section IV-A, this separation of the identifiers and locators makes it also simpler and more secure to handle mobility and multi-homing than what is currently possible.

Figure 1 shows, in approximate terms, how the new HIP sublayer is located in the current stack. On the layers above the HIP sublayer, the locator(s) of the host need not be known. Only the HI (or its 128-bit representation, a Host Identity Tag, HIT, or a 32-bit local presentation, a Local Scope Identifier, LSI) are used[11]. The Host Identity sublayer maintains mappings between identities and locators. When a mobile host changes its location, HIP is used to transfer the information to all peer hosts. The dynamic mapping from the identifier to locators, on other hosts, is modified to contain the new locator information. Upper layers, e.g. applications, can remain unaware of this change; this leads to effective division of labour and provides for backwards compatibility.

During the connection initialisation between two HIP hosts, a four-way handshake, a Base Exchange (see Section IV-D, below), is run between the hosts [28]. During the exchange, the hosts identify each other using public key cryptography and exchange Diffie-Hellman public values. Based on these values, a shared session key is generated. Further, the Diffie-Hellman key is used to generate keying material for other cryptographic operations, such as message integrity and confidentiality. During the Base Exchange, the hosts negotiate what cryptographic protocols to use to protect the signalling and data messages. As of today, the default option is to establish a pair of IPsec Encapsulated Security Payload (ESP) Security Association (SA) between the hosts [29]. The ESP keys are retrieved from the generated Diffie-Hellman key and all further user data traffic is sent as protected with the ESP SAs. However, the HIP architecture is not limited to support only ESP. With suitable signalling extensions, some in preparation [30], it is possible to use for protecting the user data almost any standalone data protection protocol, such as SRTP [31] for real-time multimedia and perhaps even data-oriented, copy-and-forward protocols such as S/MIME [22].

From an overall, functional point of view, HIP provides the same four invariants described in Section IV-A, but with additional functionality: (mobility, multihoming, authentication, encryption). However, since even today IP still provides universal – even if limited – connectivity throughout the world, the efforts on HIP have more focused on those aspects of end-to-end connectivity that today's IP does not provide that well: provisions for mobile and multi-homed hosts (see Section V-A), for baseline security [32], for middle-box support (Section V-D), and for connectivity between the two versions of IP, IPv4 and IPv6 (in Section V-C).

[11]Note that all the current HIP implementations still allow IP addresses to be used above the HIP sublayer. In that case, the host and application act as if HIP was not installed into the system.

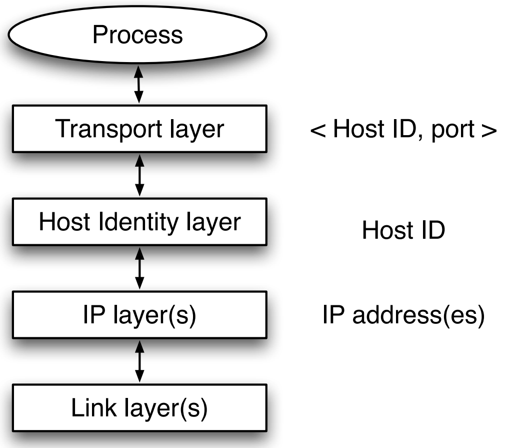

Fig. 1. Approximate location of the HIP sublayer within the TCP/IP stack.

C. HITs and LSIs

When HIP is used, the Host Identity public keys are usually not written out directly. Instead, their 128-bit long representations, Host Identity Tags (HIT), are used in most contexts. According to the current specifications [33], a HIT looks like an IPv6 address with the special 28-bit prefix 2001:0010::/28, called Orchid[12], followed by 100 bits taken from a cryptographic hash of the public key. It is important to note that embedding a cryptographic hash of the public key to the short identifier allows one to verify that a given HIT was derived from the given Host Identity. That is, due to the second pre-image resistance of the used hash functions, it is believed to be computationally unfeasible to construct a new Host Identity that hashes to a given, existing Host Identity Tag. Therefore, HITs are compact, secure handles to the public keys they represent.

As described in Sections IV-D and IV-F, below, HITs are used to identify the communication parties both in the HIP protocol itself and in the legacy IPv6 APIs. The second pre-image resistance property of the hash establishes implicit channel bindings between the HIT and the underlying IPsec or other security associations. That is, if an application uses a HIT to connect a socket, it implicitly gains assurance that once the socket connects, the sent packets will be delivered to the entity identified by the corresponding Host Identity, and any received packets indeed come from that entity.

Unfortunately, in the IPv4 API the host identifiers, i.e, IP addresses, are only 32 bits long. Hence, even if all of these 32 bits were derived from the hash of the public key, there still

[12]The Orchid acronym stands for Overlay Routable Cryptographic Hash IDentifiers.

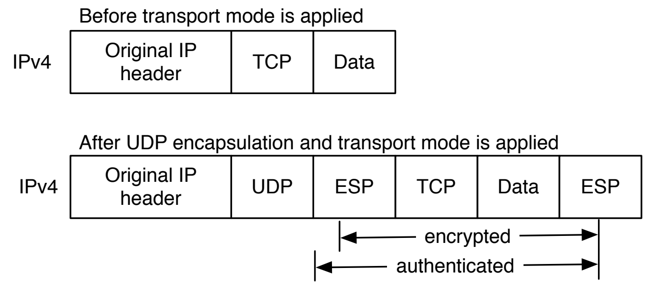

Fig. 2. IPsec NAT-Traversal.

would be occasional collisions. Hence, the approach taken by HIP is to so-called Local Scope Identifiers (LSIs) in the IPv4 API [32]. These are assumed to be only locally unique; there are also no implicit channel bindings. However, with suitable IPsec policy expression it is still possible to create explicit channel bindings even for LSIs. Unlike HITs, LSIs are not sent on the wire within the HIP protocol, and have only local scope, so they cannot be reliably used to name hosts within the network (e.g. in access control lists).

D. Protocols and packet formats

From the protocol point of view, HIP consists of a control protocol, a number of extensions to the control protocol, and any number of data protocols. The control protocol consists of the base exchange, any number of status update packets (that are typically used to convey extension protocols), and a three message termination handshake that allows the peer hosts to cleanly terminate a protocol run [28]. For most extensions, there is some flexibility which messages are used to carry the payloads comprising the extension. For example, multi-homing related information may be sent in three of the initial handshake messages or in update packets. With suitable extensions, HIP may be extended to use almost any data protocol. However, today the only defined data protocol is IPsec ESP [29].

By default, the HIP control protocol is carried directly in IPv4 and IPv6 packets, without any intervening TCP or UDP header. However, a larger fraction of the existing IPv4 NATs will not pass traffic with this protocol number through, or at best will allow only one host to communicate from behind the NAT. Therefore, work is being conducted to specify how HIP control messages may be carried in UDP packets [34]. The basic idea there is to use UDP encapsulation identical to IPsec IKE NAT traversal [35] [36]; see Figure 2.

However, the other aspects will differ, and since mere packet encapsulation is not enough to allow NATted hosts to be contacted from the Internet, the UDP encapsulation format must be accompanied by a specification for NAT traversal details. At the time of writing, the NAT traversal specification is preparing for the Working Group Last Call [34].

The HIP control protocol packet format, together with an underlying IPv4 packet, is depicted in Figure 3. The packet consists of a fixed header that is modelled after IPv6 extension headers. As the most important information, it also carries a packet type field and the sender's and receiver's HIT. The fixed header is followed by a variable number of *payloads*. The base exchange and each extension defines what payloads are needed and on what kind of HIP control messages the payloads may be carried. Most (but not all) messages also carry a cryptographic Hashed Message Authentication Code (HMAC) and a signature in the end of the packet. The former is meant for the peer host, which can use the Diffie-Hellman session keys necessary to verify the HMAC. The latter is meant for middle boxes, which typically do not have access to the Diffie-Hellman key but may well have access to the sender's public key [37]. After the base exchange, where the signature is used by the peer hosts for authentication, the signature is typically ignored by the receiver.

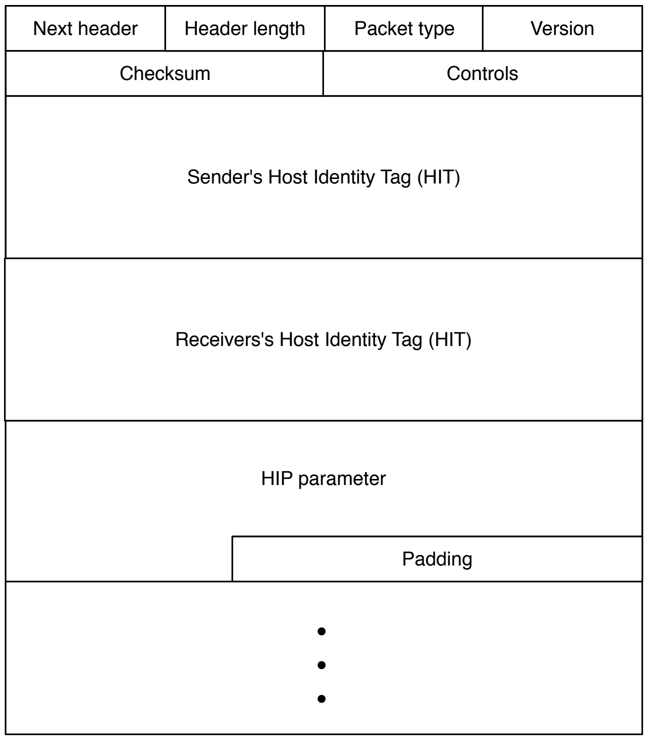

Fig. 3. HIP control packet format.

The base exchange is depicted in Figure 4. It consists of four messages, named by letters and numbers. The letters denote the sender of the packet, I for initiator or R for responder. The numbers are simply sequential. Hence, the four messages are named as I1, R1, I2, and R2. The I1 message is a mere trigger. It is used by the initiator to request an R1 message from the responder. By default, any HIP host that receives an I1 packet will blindly reply with an R1 packet; that is, the responder shall not remember the exchange.

Remaining stateless while responding to an I1 with an R1 protects the responder from state-space-exhausting denial-of-service attacks, i.e., attacks similar to the infamous TCP SYN one [38] [39]. However, as a side effect it adds flexibility to the architecture. It does not need to be the responder itself that replies to an I1. Hence, if there is some other means by which the initiator may acquire a fresh R1 message, such as a directory look up, it is perfectly fine to skip the I1/R1 exchange. As long as the host responding with an R1 has a supply of fresh R1s from the responder, it can be any node. This flexibility is used by some more advanced architecture

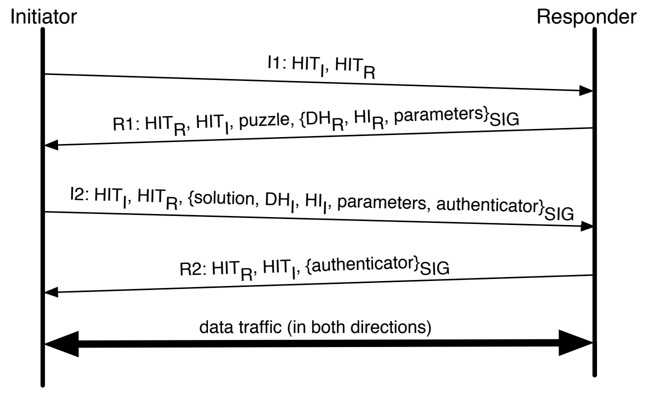
Initiator
Responder
I1: HIT_I, HIT_R
R1: HIT_R, HIT_I, puzzle, {DH_R, HI_R, parameters}_SIG
I2: HIT_I, HIT_R, {solution, DH_I, HI_I, parameters, authenticator}_SIG
R2: HIT_R, HIT_I, {authenticator}_SIG
data traffic (in both directions)

Fig. 4. HIP base exchange.

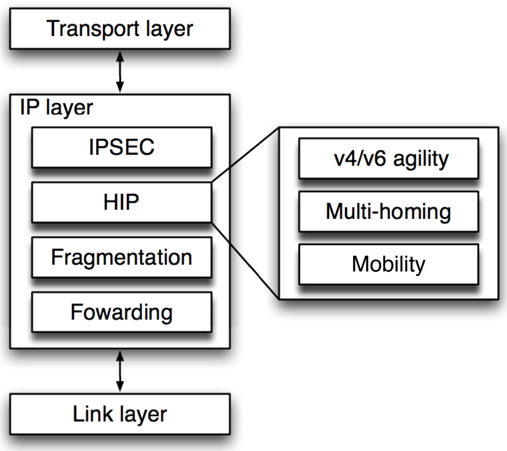

Fig. 5. New layering, with the HIP sublayer, in detail.

proposals basing on HIP, such as the Hi3 proposal [40]; see Section VI-B.

The R1 message contains a cryptographic puzzle, a public Diffie-Hellman key, and the responder's public Host Identity key. The Diffie-Hellman key in the R1 message allows the initiator to compute the Diffie-Hellman session key. Hence, when constructing the I2 message it already has the session key and can use keys derived from it.

In order to continue with the base exchange, the initiator has to solve the puzzle and supply the solution back to the responder in the I2 message. The purpose of this apparently resource-wasting method is to protect the responder from CPU-exhausting denial-of-service attacks by enforcing the initiator to spend CPU to solve the puzzle. Given the puzzle solution, the responder can, with very little effort, make sure that the puzzle has been recently generated by itself and that is has been, with high probability, solved by the initiator and is not a result of a puzzle posted much earlier or a puzzle generated by someone else. That is, by verifying the puzzle solution the responder knows that, with high probability, the initiator has indeed used quite a lot of CPU to solve the puzzle. This, seemingly, is enough to show the initiator's commitment to the communication, thereby warranting the forthcoming CPU cycles that the responder needs to process the rest of the I2 message. The difficulty of the puzzle can be varied depending on the load of the responder. For example, if the responder suspects an attack, it can post harder puzzles, thereby limiting its load.

The I2 message is the main message in the protocol. Along with the puzzle solution, it contains the initiator's public Diffie-Hellman key, the initiators public Host Identity key, optionally encrypted with the Diffie-Hellman key, and an authenticator showing that the I2 message has been recently constructed by the initiator.

Once the responder has verified the puzzle, it confidently can continue to construct the Diffie-Hellman session key, to decrypt the initiator's Host Identity public key (if encrypted), and to verify the authenticator. If the verification succeeds, the responder knows that there is out there a host that has access to the private key corresponding to the initiators Host Identity public key, that the host wants to initiate a HIP association with it, and that the two hosts share a Diffie-Hellman session key that no other node knows (unless one of the hosts has divulged it) [41]. Given this information, the responder can consult its policy database to determine if it wants to accept the HIP association or not. If it does, the responder computes an authenticator and sends it as the R2 packet to the initiator.

The details of this relatively complex cryptographic protocol are defined in the HIP based exchange specification [28]. From the high level point of view, the HIP protocol can be considered as a member of the SIGMA family [42] of key exchange protocols.

E. Detailed layering

Let us now focus in more detail how the new HIP sub-layer is wedged into the existing stack. Figure 5 depicts the positioning of the new functionality in detail. The current IP layer functionality is divided into those functions that are more end-to-end (or end-to-middle) in nature, such as IPsec, and those that are more hop-by-hop in nature, such as the actual forwarding of datagrams. HIP is injected between these two sets: architecturally immediately below IPsec, in practise often functionally embedded within the IPsec SA processing [43].

Now, in a communications system the main function of packet identifiers is to allow demultiplexing. Forwarding nodes, such as routers, use the identifiers to determine which of the outgoing links to forward the packet to. The receiving host uses the identifiers to make sure that the packet has reached its right destination and to determine which upper layer protocol (if any) should process the packet. In the classical IP system, the IP address is used both by the routers to determine the next outgoing link and by the destination system to make sure that the packet has reached the right end host. With HIP, the separation of the location and identity information disentangles these two functions. Routers continue to use IP addresses to make their forwarding decisions.

At the hosts the behaviour changes. For HIP control packets, the source and destination HIT fields in the packet determine the right processing context. For data packets, the receiving host identifies (and verifies) the correct HIP association indirectly, typically by first getting the correct session keys based on the ESP Security Parameter Index (SPI) in the received packet, and then decrypting and verifying the integrity of the packet. Thus, the actual IP addresses that were used for routing the packet are irrelevant after the packet has reached the destination interface, although they may still be used in some local access control filters if desired.

This is in stark contrast with the prevailing IP practice, where the transport layer identifiers are created by concatenating the IP-layer identifiers (IP addresses) and the port numbers. The main benefit of the current practice is implied security: since the transport identifiers are bound to the actual network locations, the transport connections get automatically bound to the locations. That allows the routing and forwarding system to be used as a weak form of security: binding identity to the location allows reachability to be used as a (weak) proxy for the identity. When HIP is used, this weak-security-by-concatenation is replaced by strong cryptographic security, based on the public cryptographic host identity keys.

F. Functional model

We now consider what happens underneath the applications, in the API, kernel, and network, when a typical, existing legacy application is configured to use HIP. Naturally, before anything HIP-related can happen, HIP must be installed into the system and the application(s) must be configured to use it. Today, typically the installation requires that a pre-compiled HIP package is installed; in most operating systems this requires administrator privileges. As the second step, the applications must be configured to use HIP. In most cases, there are three alternative configuration options. The simplest but least generic way is to configure the HITs of the peer hosts directly into the application. For example, an IPv6-capable e-mail application can be configured to use HIP by entering the mail server's HIT into the configuration field that usually contains either a DNS name or an IP address. An IPv4-only application could be similarly configured with an LSI if the system is configured to securely map the LSI to a unique HIT [32].

A more transparent way is to change the mapping from DNS names to IP addresses in a way that resolving a DNS name returns a HIT (or an LSI), instead of an IP address, to the application. Obviously, there are several options how to implement that. In most UNIX-based systems the simplest way to change the mapping is to modify the local /etc/hosts file. Another, non-standard way is to store the HIT or LSI into the DNS in an AAAA or an A record. The drawback of this method is that as a result of such practise non-HIP-aware hosts may fail in non-obvious ways. Finally, the standard way is to store the HIT (and other information) into the new HIP resource record [44]. That allows both HIP and non-HIP hosts to create new connections with the target host without new difficulties.

Once the application has got a HIT (or an LSI), it uses it in various socket API calls. In a typical implementation, the underlying libraries and the communication stack handles the HIT just as if it were a vanilla IPv6 address, all the way until the resulting packet is delivered to the IPsec module for processing. At the IPsec level, an IPsec policy rule is used to detect that the destination and source IP address fields in the packet contain the Orchid prefix. (For LSIs, an explicit, LSI-specific rule is typically required). Typically, all Orchid packets are passed to IPsec ESP for processing[13]. If there are no ESP Security Associations yet, the IPsec module requests a suitable pair of security associations from the HIP control functionality, which in turn creates the SAs by executing the HIP ESP extension [29], either as a part of a base exchange or over an existing HIP control association using update messages.

For ESP processing, typical HIP implementations use a non-standard variant of the ESP modes, called the BEET mode [43]. The BEET mode can be considered as standard transport mode that is enhanced with build-in address rewriting capabilities. To achieve HIP functionality, at the sending end the SA is configured so that the HITs in an outgoing packet are converted to IP addresses. Conversely, at the receiving end the IP addresses in the packet are discarded and, if the packet passes integrity verification, the HITs are placed in the packet header. As a part of this processing, it is also possible to rewrite the IPv6 header used to carry the HITs (or the IPv4 header carrying LSIs) into an IPv4 header carrying IPv4 addresses (resp. IPv6 header carrying IPv6 addresses). Between the sender and the receiver, the packet looks like a standard IPsec ESP transport mode packet, with IP addresses in the header, and is handled by all HIP-unaware nodes as such.

At the receiving end, once an incoming packet has been processed by the IPsec ESP module, the packet contains the sender's HIT in its source field. Since this HIT was placed to the packet during the IPsec processing, and only if the packet passed verification, the HIP sublayer provides assurance to all of the upper layers that the packet was indeed received through an IPsec Security Association that was securely created through a cryptographic protocol where the private key corresponding to the HIT was present. In other words, the upper layers at the receiver end can trust in that the Host Identity represented by the source address is indeed valid and not a result of IP source address spoofing[14].

G. Managing the HIP name space

HIP adds another name space to the existing name spaces in the Internet. Users and developers already deal with names such as SIP URIs and DNS names; these names are structured, human friendly, and managed by some authority (e.g. a network administrator). Protocols often deal with IP addresses; again, these are structured, centrally allocated names. HIP names are quite different; they are unstructured (keys or hashes of keys), not human friendly, and not necessarily managed by anyone (keys may be self generated). Moreover, the bindings between HIP names and other names must be secured somehow, if other names are used in the overall system. We do not expect that humans will want to handle HITs. A variety of mechanisms are available to bind other

[13]Note that if the source and destination addresses are other than Orchids, the packet is processed as if HIP did not exist in the system at all. In that way, all HIP enabled systems remain fully backwards compatible with non-HIP-enabled systems.

[14]Obviously, for this to work the IPsec SPD needs to contain a rule that discards any incoming, unprotected traffic containing the Orchid prefix in the source or destination address.

names to HITs, including DNSSEC, SPKI/SDSI, and X.509 certificates. These techniques require some local trust anchors in the system to secure the bindings between the other names and HITs.

For example, users or applications may use DNS names to refer to a server, and may not want to directly handle HITs. Therefore, the system must be able to (securely) map between the DNS name and a HIT that belongs to the named host. HIP has defined resource record types that allow for a host identity to be stored in the DNS and indexed by domain name. DNSSEC is a technique to provide data integrity and data origin authentication in such a scenario. Presently, because HITs are flat and hard to store in the hierarchical DNS, a user or application starting with a DNS name must fetch both the host identity and the IP address in separate calls to DNS. Applications or stacks that wait for DNS HIP records to be checked may impose additional latency on the session initiation. Furthermore, an application starting with a HIT cannot use DNS to find IP addresses. Instead, researchers have explored the use of distributed hash tables to store name records indexed by HITs. If more than one such HIT-based name service is deployed, an additional wrinkle is that it is not possible to learn of the particular name service(s) in use by the host simply by inspecting the host's HIT. These issues are currently being worked in the IETF HIP working group. A preliminary study concluded that a name resolution system that scales up to millions of mobile hosts can be constructed for HIP.

Another cost is that HIP imposes a new management load on hosts and enterprises (when identifiers are centrally managed) to manage this additional name space. Key management needs to be more carefully considered and worked out. Another consideration is that HITs cannot be aggregated; this makes it difficult to build access control lists with HIP names. Also, an additional level of indirection may cause an increase in hard-to-debug network configuration errors and failures, which current implementations are only beginning to address adequately.

H. Other costs

While HIP has generally been carefully designed to be backwards compatible with existing applications and infrastructure, obviously any change may have its drawbacks or costs; so too with HIP. In this section we briefly discuss the most general potential costs; there certainly are others, more situation-specific ones.

Perhaps the biggest difference to present communication that HIP introduces is a slight delay, caused by the base exchange, whenever starting to communicate with a new host. This delay is mainly caused by the time taken to solve the puzzle, to process the public key signatures, and to produce the Diffie-Hellman public key. While the amount of time can be somewhat reduced by using trivial puzzles and short keys, it cannot be eliminated. One potential way to alleviate the situation to use Lightweight HIP (LHIP), a variant of HIP proposed by Heer [45]. However, both of these options are clearly less secure than the baseline HIP.

A second, often mentioned drawback is the need to change the operating system kernel; many people are understandably concerned about this, even though the modifications are very small, as discussed in Section IV-E. One alternative, used by the Boeing implementation (see Section VII-A), is to divert the traffic to user level and process all packets there. Although this alternative has been shown to perform well for typical Internet usage, kernel-based implementations are preferred for high-performance, high-bandwidth usage. A complete patch for the Bound End-to-End Tunnel (BEET) mode including inter-family communication had been accepted to the official Linux kernel starting from version 2.6.27.

An often cited potential drawback with HIP relates to the so-called third party referral problem, where one host sends a name of a second host to a third host. In practise, in a third party referral situation the names are IP addresses and there are three IP hosts, A, B, and C. Hosts A and B have an ongoing connection (such as a TCP connection); therefore A knows B's IP address. Host A now wants to initiate a connection between B and C by telling C to contact B. Obviously, it does that by sending Bs IP address to C. With the IP address at hand, C is now able to open a new connection to B. Now, as the HITs are not routable, it is hard to open a new HIP association to a HIP host if all that one has is a HIT. Hence, if the third-party-referral application is a legacy one and if it imagines to use IP addresses and uses HITs (or LSIs) instead, the referral process may fail. We allege that the third party referral problem is not that important nor bad in practise. Firstly, in a NATted environment, third party referrals fail already now, indicating that the problem may be less important than what is commonly claimed. That is, most commonly used applications no longer rely on third party referral working. Secondly, using an overlay network to route HITs (see Section VI-B), it is possible to support even legacy applications that rely on third party referrals working.

There are some considerations related to how HIP will work in a host where IPsec is used also for other, non-HIP purposes. For example, both HIP and IPsec use the same SPI space and SPI selection must be coordinated. As another example, it remains an open question whether it is possible to first convert HITs into IP addresses, then run these IP addresses over an IPsec VPN connection. While these problems are real, HIP is not alone with them. The current IPsec architecture specification [5] is not too specific in explaining how a host should behave in a situation where IPsec is applied repeatedly.

Finally, some diagnostic applications, and probably a few other specialized ones, will not work with HIP, at least not as intended. For example, the diagnostic tool ping can be used with HIP, but when given a HIT it no longer tests IP connectivity but HIP-based connectivity. Similar surprises are likely to be detected with other diagnostic applications. However, the fact that HITs and other IP addresses are clearly distinguishable by the ORCHID prefix, we doubt whether these new failure models would hamper operations in practise[15].

V. Mobility, multi-homing, and connectivity

Equipped with a basic understanding of what HIP is and how it works, we now continue to study how it, together

[15]At the time of writing, the use of HIP underneath an application that does STUN/ICE has not been tested yet.

with a number of defined and prospective extensions, can be used to address the problems discussed in Section III. In this section, we first discuss basic host mobility and multi-homing (Section V-A) and rendezvous (Section V-B), and then connectivity related issues including locator agility (Section V-C), architected NAT traversal (Section V-D), subnetwork mobility (Section V-E), and application-level mobility (Section V-F). The problems related to unwanted traffic, privacy, and accountability are left to Section VI.

A. *HIP-based basic mobility and multi-homing*

We first describe how HIP mobility and multi-homing are designed to work in a non-NATted environment, and later return to the NAT traversal design in Section V-C. As discussed above, with HIP packet identification and routing can be cleanly separated from each other. A host receiving a HIP control packet (other than I1) can verify its origin by verifying the packet signature; alternatively, the two endpoints of an active HIP association can simply verify the message authentication code. A host receiving a data packet can securely identify the sender through a three step process: it first locates an ESP Security Association based on the Security Parameter Index (SPI) carried in the packet. As the second step, it verifies packet integrity and then decrypts the packet, if needed. Finally, as the third step, it places into the packet the source and destination HITs, as stored within the ESP BEET-mode SA. Thus, the actual IP addresses that were used for routing the packet are irrelevant after the packet has reached the destination interface.

Hence, to support mobility and multi-homing with HIP, all that is needed is the ability of controlling what IP addresses are placed in outgoing packets. As the addresses will be ignored by the recipient in any case, the sender may change the source address at will; for the destination address, however, it must know the address or addresses that the receiver is currently being able to receive packets at.

The HIP mobility and multi-homing extension [46], [47] defines a Locator parameter that contains the current IP address(es) of the sending entity. For example, when the mobile host changes its location and therefore IP address, it generates a HIP control packet with one or more Locator parameters, protects the packets integrity, and sends the packet to its currently active peer hosts. Note that the IP version of the locators may vary; it is even possible to use both IPv4 and IPv6 addresses simultaneously, and make a decision of the IP version used on outgoing IP packets depending on a local policy.

When the host receives a Locator parameter over an active HIP association, it needs to verify the reachability of the IP address(es) that are included in the parameter. Reachability verification is needed to avoid accepting non-functional and falsified updates[16]. The verification can be skipped in special circumstances, for example, when the peer host knows that the network screens all address updates and passes only valid ones.

[16]Note that the peer host does not need to be completely trusted. Hence, it is plausible that it may lie, for example, to launch a flooding attack [48].

The HIP mobility and multi-homing specification [46] describes also an enhancement to the standard location update procedure. The Credit-Based Authorisation (CBA) process allows the peer host to use the new locator before the reachability test has passed. To prevent flooding attacks, the peer host calculates *a credit* based on the amount of data received from the mobile host. When sending to one or more addresses whose reachability status is unknown, the host may send, at maximum, as much data as it has previously received from the mobile host, i.e., up to the credit limit. Naturally, this limitation is lifted as soon as the reachability status of the address in question becomes verified. This method can be used to enhance performance of certain real-time applications. For example, voice-over-IP applications do suffer from long breaks in a connection; with this method, the break can be made much shorter or eliminated.

When handovers are made in the so-called break-before-make manner, all connectivity can be lost for a while. HIP supports also make-before-break style handovers, enhancing the handover performance significantly while the handover can be made even without any packet loss.

B. *Facilitating rendezvous*

While the mobility and multi-homing extension specifies how two hosts that already have a HIP association can exchange locator information and change the destination address in outgoing traffic, there remains another mobility-related problem: rendezvous. When another host wants to make a contact with a mobile host, when two mobile hosts have moved simultaneously and have both stale peer address information, or whenever the destination hosts current IP address is unknown, e.g. since it is dynamic or kept private (see Section VI-B), there needs to be an external means to send packets to the host whose current locator is not known. In the Mobile IP world, this function is provided by the Home Agent, which forwards any messages sent to the mobile hosts home address to the mobile host itself.

In the HIP mobility architecture a similar function is provided by a Rendezvous server. Like a Mobile IP home agent, a HIP Rendezvous server tracks the IP addresses at which hosts are reachable and forwards packets received thereto. Unlike a home agent, a HIP rendezvous server forwards only HIP signalling packets (by default only the first packet of a base exchange), and the rest of the base exchange, as well as all subsequent communications, proceed over a direct path between the host and its peer. Furthermore, a HIP host may have more than one rendezvous server; it may also dynamically change the set of rendezvous servers that is able to serve it.

In HIP terms, a Rendezvous Server is simply a HIP host that knows another HIP host's current locator and is willing to forward I1 packets (and possibly other HIP control packets) to that host. In theory, any HIP host could act as a rendezvous server for any of its active peer hosts, as it already knows the peer host's locator or locators. However, in practical terms it is expected that there will be a number of stationary hosts, located in the public Internet, providing rendezvous as a service.

The rendezvous service is defined by means of two HIP extensions. First, the generic service registration extension [49] is used by a rendezvous server and a prospective client to agree on the existence and usage of the service in the first place. Second, the rendezvous service extension [50] defines the terms of the specific service. Both of the specifications are quite simple and straightforward. The reason there are two documents, and not only one, is reusability: the registration extension can be used to define also other services besides the rendezvous service.

The HIP rendezvous server is very simple. When processing incoming HIP control packets, if the server receives a packet that does not contain any of its own HITs, the server consults its current HIT-to-locator mapping table. If there is a match, the packet is forwarded to the locator listed in the table. To facilitate tracking, the packet is augmented with the original addresses from the incoming packet. However, what is noteworthy here is that the IP addresses in the incoming packet are not used for any demultiplexing decisions; they are simply copied to a new HIP parameter, which is then added to the packet. Hence, the rendezvous server does not need a pool of IP addresses, in the way a Mobile IP Home Agent needs. As each mobile host is identified by the HIT and not the address, the destination address in the incoming packet is transient.

C. Mobility between addressing realms and through NATs

As described earlier, HIP supports mobility between and within different IP address realms, i.e., both within and between IPv4 and IPv6 networks. However, all addresses in the scenarios above have been assumed to be drawn from the public address base, i.e., we have implicitly assumed that the (functional) addresses in the Locator payload are routable. As of today, the mobility and multi-homing specification [46] does not describe how one can be mobile when the host has addresses only from a private address space, i.e. when it is behind a legacy NAT device. However, there is ongoing work to define how NATs can be enhanced to understand HIP and how HIP can be enhanced to pass legacy NATs.

From a HIP point of view, there may be two types of NATs: HIP-unaware legacy NATs, and HIP-aware NATs. We first consider the current engineering work to address the legacy problem, and then turn our attention to how to engineer NATs into the overall architecture.

At the time of this writing, the HIP Working Group at the IETF seems to be converging on a solution for how HIP should work over legacy NATs. The main requirement is that the HIP control packets and ESP protected data traffic should be carried embedded in UDP packets in the same way IPsec IKE and ESP traffic is currently embedded in the IPsec NAT traversal solution [36]. There seems also to be agreement that to initiate sessions through a NAT, a HIP relay server (similar to the rendezvous server) can be used to forward initial packets through the NATs and that the ICE [6] methodology (originally designed for SIP) will be used to collect the so-called candidate address sets and to pick the most appropriate address pair for the actual control and data traffic. It seems inevitable that ICE will need, at minimum, some small modifications as in the HIP case; the UDP packets will carry ESP, which in turn carry the actual data traffic, while in the case of SIP-related ICE use, the UDP packets may also carry plain data traffic[17]. In some environments, it may be desirable to use plain ESP instead of UDP-encapsulated ESP in the case there are no NATs on the path or the NATs happen to support ESP. However, such an option may also have problems running over firewalls even in non-NATted environments, and is likely to complicate the address pair selection algorithms somewhat.

After some discussion, the design team working on HIP NAT traversal agreed to reuse ICE as closely as it can be reused from its current support for SIP; i.e., through using STUN [51] and TURN [52] to collect the candidate addresses and assist in relaying, and to use STUN to perform the address probing [53]. Some observers, including the lead author of this article, argued that the solution should use ICE methodology but encode all information in HIP parameters, i.e., integrated ICE tightly to HIP. That is, instead of using STUN and TURN, the HIP rendezvous servers would offer services similar to the service provided by STUN and TURN [34]. According to our initial analysis, it will be simpler, both in the terms of exact specifications and especially for implementation, to follow the latter path. In the former case, the TURN server, HIP rendezvous server, and the HIP host side implementation all need to be modified; at the host side integrating the STUN/TURN library to a HIP daemon can be a quite complex task. In the latter case, it suffices to modify the HIP rendezvous server and the HIP host side implementation. Furthermore, these modifications appear relatively simple.

Anyway, the design seems to have converged now on reusing ICE, STUN, and TURN as closely as possible. Once the candidate address collection and address pair selection process is there and the candidate addresses are exchanged in HIP Locator parameters, the whole matter of mobility with NAT traversal becomes trivial. To implement that, all that is needed is for the mobile host to collect candidate addresses after each movement, to send the addresses to its peer hosts (using already existing mechanisms), and probe what address pair is the best one, using the already existing mechanisms, enhanced with algorithmic wisdom from ICE.

D. Architected NAT traversal

We now turn out attention into how to make NATs an integral part of the architecture. In contrast to the legacy NATs, whose traversal mechanism could be viewed as something of a hack as it requires both UDP encapsulation and explicit support nodes at the un-NATted side of the network, integrated NATs are architecturally cleaner; they pass all data protocols, including ESP, and do not require external support nodes. One such approach is presented in the SPINAT proposal [54]. A SPINAT-based NAT device is HIP-aware and can take advantage of the passing-by HIP control packets, using them to learn the necessary details for demultiplexing data protocols. Additionally, a SPINAT device must also implement the HIP

[17]There are a few specific issues here: 1) need for a method to signal the use of ESP, 2) a decision on how to handle upper layer checksums, as NATs are unable to rewrite them within ESP encapsulation, and 3) if SIP is used on the top of HIP, the SIP messages will most probably need to carry HITs in addition to IP addresses.

rendezvous functionality, acting as a rendezvous server for all hosts that are behind it.

In practice, a SPINAT device uses information from the HIP Base Exchange and UPDATE packets to determine the IP addresses, HITs, and SPI values (or other contextual identifiers) in the data protocol[18]. With this information, the SPINAT device can do required address translations between public and private address spaces, mapping several HIP identities (and corresponding data protocols) to a single external IP address.

With SPINAT, a HIP host residing behind it learns directly the mapped address as a part of rendezvous registration. Furthermore, there is no difference between a mapped address, as identified by STUN, and an allocated address, as offered by TURN. They are both replaced by one shared external address, the data protocols being explicitly mapped with the help of the information from the HIP control messages.

E. Subnetwork mobility

So far we have only considered mobility in the terms of mobile hosts. Now we turn our attention towards mobility at other granularity levels, first considering how HIP can be used to implement subnetwork mobility and then, in the next subsection, how application-level mobility can be implemented. A key feature to both of these is delegation. That is, the use of public cryptographic keys as identifiers adds cryptographically secure *delegation*, as a primary element, into the architecture. That, in turn, can be used to implement different forms of proxied functionality; for example, subnet mobility, allowing an intelligent network to signal mobility on mobile hosts or subnets behalf, and allowing application level service delegation.

The basic idea behind cryptographic delegation is simple, but the result is powerful [55]. When a principal (e.g., a host), identified by a public key, wants to delegate some rights (such as access rights) to another principal, identified by another public key, all that is needed that the former signs a statement indicating that the latter is authorised to perform the operations needing the rights. The statement must include the delegate's public key to identify the delegate and it must be properly signed by the delegator's private key so that its validity can be verified. Furthermore, the delegator must itself possess the delegated right. Typically, but not necessarily, such delegations form an implicit loop where authority flows from a resource possessor through some intermediates to the prospective resource consumer, and from there back to the resource owner [55].

In the context of mobility and mobile sub-networks, delegation can be used to delegate the actual act of sending HIP control messages that contain new locators from the individual mobile hosts to a mobile router, and further from the mobile router to some infrastructure node at the fixed network side.

Figure 6 illustrates the basic idea. First, individual mobile hosts in a mobile network delegate, to a mobile router, the right to inform their peer hosts about their location. At this

[18]IP addresses are needed to identify the context. HITs are needed to forward HIP signalling packets, such as Updates. SPIs are needed to forward ESP-protected data packets. When other data protocols but ESP are used, there needs to be other sufficient context-setting identifiers to allow forwarding to work.

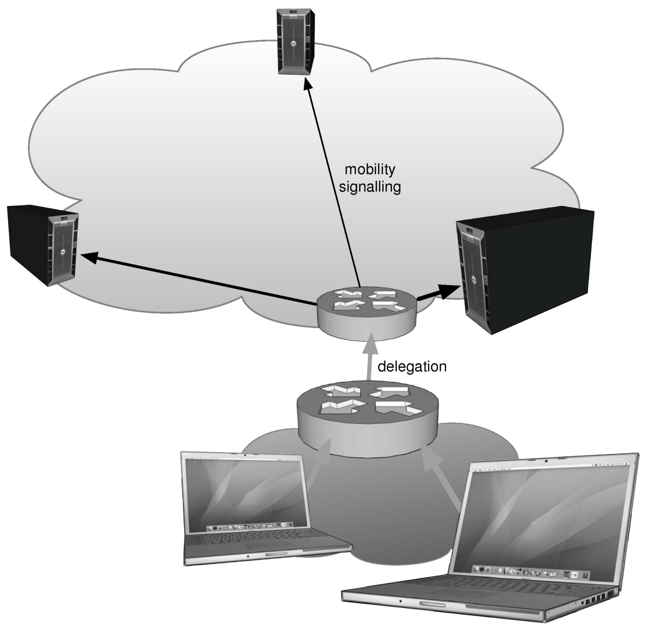

Fig. 6. A moving network scenario.

stage the mobile router could send HIP signalling messages on the behalf of all the hosts behind it, but such a practise would use quite much capacity at the air interface. Hence, in the next step, the mobile router further delegates that right to a router (or another infrastructure node) within the fixed part of the network (illustrated as a cloud). Once the fixed side router learns that the mobile subnetwork has moved, it will send Updates to the relevant peer hosts.

As an additional optimisation, if the underlying IP-layer router mobility functionality is arranged in such a way that the fixed-side router gets directly informed whenever the mobile router changes its point of attachment, it becomes possible for the fixed-side router to send the mobility messages directly to the corresponding hosts of all mobile hosts within the mobile subnetwork, without any signalling at all over the air interface. Hence, for HIP-based mobility signalling, no HIP-related messages need to be transmitted over radio at mobility events. On the other hand, whenever a new host joins the mobile network, a few messages are needed to establish the delegation.

F. Application-level mobility

Another area where delegation can be applied to are applications and services [56]. As illustrated in Figure 7, Host Identities can be allocated to abstract services (leftmost box) and service instances in addition to physical hosts. With that kind of arrangement, using suitable service resolution infrastructure, a client application can ask for a connection to the abstract service, using the HIT assigned to the abstract service and get delegated and redirected to one of the service instances. Furthermore, for host mobility, the signalling right for mobility signalling can be further delegated from the service instances to the physical node,thereby allowing host mobility to securely update the clients understanding of the locations of the service instances.

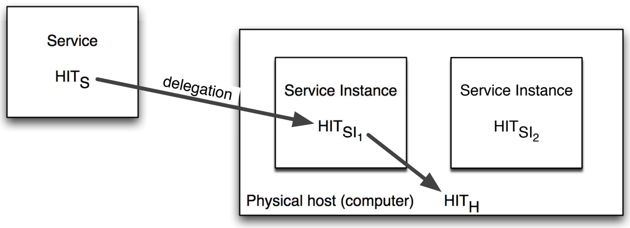

Fig. 7. Using abstract hosts as a base for service-level names.

VI. Privacy, accountability, and unwanted traffic

A. Privacy and accountability

The base use of public cryptographic keys for host identification is clearly a means that enhances accountability. At the same time, it can endanger privacy. HIP tries to approach a balance, allowing both enhanced accountability and privacy. Naturally, achieving such a balance is tricky, and it remains to be seen whether the mechanisms currently built-in and proposed are sufficient. In HIP, the Host Identity public keys may be anonymous in the sense they do not need to be registered anywhere. For example, a privacy conscious host could create a multitude of key pairs and identify itself through a different key to each web site during a surfing session. Were this combined with dynamically changing IP addresses, the web sites could not correlate the activity through lower layer identifiers; however, anything identity-revealing in HTTP, such as cookies, could still be used, of course.

Unfortunately, while using multiple identities and changing addresses allows one to preserve privacy towards remote peer hosts, that is not sufficient to remain anonymous or pseudonymous towards the network. The network nodes close to the host could easily correlate the public keys over multiple IP addresses, determining which traffic flows belong to which host, with high probability. Encrypting the Host Identity key in the I2 packet is not sufficient to defeat such tracking, as the HITs, present in all control packets, can be easily used instead.

To alleviate the situation, a few years ago we proposed a so-called BLIND extension to HIP [25]. The basic idea is simple: instead of sending plain HITs in control packets, one hashes the HIT with a random number and sends the hash result and the random number in the initial control packets. Once the connection has been established, the actual HIT can be revealed to the responder. Alternatively, if the responder has only a limited number of HITs that it accepts connections from, it can try each of them in turn to see if the incoming connection is from a trusted peer host.

The BLIND approach was implemented and simultaneously enhanced by Takkinen [26]. Besides using BLIND, the approach also uses identifier sequences [57]. That is, they replace all constant, easily traceable identifiers, in the HIP control protocol and in the data protocol below the encryption layer, with pseudo-random sequences of identifiers. The peer hosts derive the seeds for the pseudo-random number generator from the HIP Diffie-Hellman session key. While the exact details of their approach fall beyond the scope of this paper, the result appears to provide a high level of privacy towards all eavesdropping third parties while still allowing the peer hosts to communicate efficiently and securely.

Altogether, it looks like that while the exact details of a comprehensive, HIP-based privacy and accountability approach remains to be defined, the pieces of existing work clearly indicate that it is possible to simultaneously enhance both privacy and accountability through clever use of HIP-based mechanisms.

B. Reducing unwanted traffic

As we discussed in Section III-D, there are two opportunities in the HIP design that allow the fundamental root causes of unwanted traffic to be affected. Firstly, we can change the architecture so that the recipient names are either not immediately accessible to the prospective senders or by requiring the recipient's consent before the network delivers any packet to the recipient. Secondly, we can attempt to raise the marginal cost of sending packets or reduce the marginal cost of receiving packets.

In the HIP context, we can consider the base exchange to employ the latter approach to defeat state space and CPU exhausting denial of service attacks. Using the former approach requires more changes; one possible way might be what we have formerly proposed in the Hi3 overlay architecture [58]. The basic idea is to hide recipients IP addresses and to require explicit consent from the recipients before a sender can use the addresses as destinations for sending traffic to.

From an architectural point of view, overlays similar to Hi3 create an additional routing layer on top of the IP layer. Other overlays aiming at added protection, such as SOS by Keromytis et al. [59] or k-anonymous overlays by Wang et al. [60], work basically in the same way.

An important aspect in overlay networks is that they change the naming structure. In more primitive cases, they merely replace the current IP addressing structure with another destination-oriented name space[19]. However, at the same time they may make denial of service attacks harder by dispersing the interface, i.e., instead of choking a single target host the potential attacker must now flood the whole overlay system, which may consist of thousands or millions of nodes. That increases the overall cost of sending by introducing artificial costs to force the participating nodes to play by the rules. The more advanced overlays further change the rules by changing the naming focus from hosts and locations to pieces of information.

The basic Hi3 setting is illustrated in Figure 8. The network consists of two planes, a data plane that is a plain IP-based router network, with HIP-enabled firewalls located at strategic points, and a control plane that is implemented as an overlay on top of the data plane. In practise, the control plane consists of enhanced HIP rendezvous servers that typically synchronise location information either partially or fully between each other.

[19]With destination-oriented name spaces we denote architectural practises where the destinations are identified. In the more advanced overlay structures the name space may become sender-oriented, e.g., along the recent publish/subscribe thinking. However, such practises fall beyond the scope of this paper.

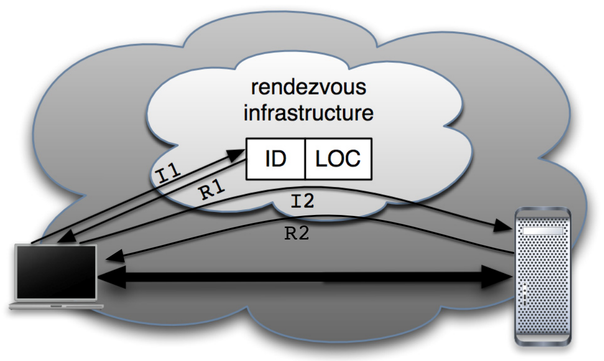

Fig. 8. Hi3 architecture.

As should be apparent by now, HIP-enabled firewalls can authenticate passing HIP base exchange and update packets, and punch holes for IPsec ESP traffic selectively [58]. In the Hi3 architecture all servers are located behind HIP-enabled firewalls. To become accessible, the servers must register to the rendezvous infrastructure, creating a binding between the server's identity (ID) and the prospective locators (R). While registering, the servers may also cache a number of pre-computed R1 packets at the rendezvous infrastructure.

When a client wants to make a contact with a server, it sends the first HIP base exchange message, I1, as usual. However, this packet cannot be sent to the server, for two reasons. Firstly, the client will not know the server's IP address. Secondly, even if it knew one, an intervening firewall would drop the packet. Hence, the only option is to send the packet to the rendezvous infrastructure. The infrastructure looks up a cached R1 packet, if it has one, and passes it to the client (otherwise it needs to pass the packet to the server, which is undesirable). The client solves the puzzle, in the usual way, and sends an I2 packet, again to the rendezvous infrastructure. The receiving node within the infrastructure verifies that the puzzle has been correctly solved and, if so, passes the I2 packet to the server. The firewall will pass the packet as it is coming from the infrastructure. At this point, the server can verify the clients identity and determine if the client is authorised to establish a HIP association. Hence, if and only if the client can be positively identified and has proper authority, the server responds to the client with an R2 packet. The R2 packet can either be sent directly to the client if it will pass the firewalls anyway, or it may be necessary to direct it through the infrastructure, thereby indirectly triggering hole punching at the firewall. Finally, the actual data traffic traverses directly through the data plane, through the firewalls.

The result of this arrangement is that only authorised clients will ever learn the server's IP address. Furthermore, if an IP address is still revealed and used to launch a traffic-based denial-of-service attack against the server or its network, the firewall will stop most of the traffic, as the packets would be arriving with unauthorised packet identifiers. In case the firewall itself becomes congested, remaining legitimate traffic can be redirected through other firewalls, using the HIP mobility and multi-homing extension.

In summary, the Hi3 proposal introduces one particular way to make certain types of denial of service attacks harder than they are today, by making IP addresses less accessible, without compromising legitimate connectivity or data traffic efficiency.

VII. Maturity status

So far we have described the HIP architecture, basic design, a number of extensions, and discussed how they can or possibly could be used to alleviate a number of hard problems in the present Internet. We now turn our focus more to the present, and describe the current standardisation and implementation status. All basic research and development for the first usable version of HIP is ready. There are three open source implementations, by Ericsson Research Nomadic Lab, Helsinki Institute for Information Technology (HIIT) [61], and the OpenHIP project. These all are mature for experimental use, differing on used platforms and supported extensions. Today, HIP is used by a few people on a daily basis, and in daily production use at one Boeing airplane assembly factory. RFCs 5201-5207 were published during 2008 and HIP implementations were updated according to the specifications.

A. Usage of HIP today

Individual researchers at Ericsson, HIIT, and Boeing use HIP in their everyday life, mostly using Linux laptops to access specific HIP-enabled services, such as e-mail. Most of their traffic still flows over vanilla IPv4, though, with HIP being used only for the few services where the servers are HIP-enabled, too.

Two major government organisations in Europe are seriously considering adopting HIP for their internal use[20]. However, at the time of writing (Spring 2009), neither of them have made their decisions. A number of additional, mainly governmental organisations have shown initial interest. The main reason for these organisations being interested seems to be the combination of baseline security and flexible mobility, where the mobility support allows one to use different access networks at the same time.

The Boeing Company has been experimenting with HIP as a component of an overall Secure Mobile Architecture (SMA) [62] in its enterprise network. Boeing's SMA implementation integrates HIP with the company's public-key infrastructure (PKI) and Lightweight Directory Access Protocol (LDAP) back-end, as well as with location-enabled network services (LENS) that can identify the location of a wireless device through triangulation and signal strength measurements [63]. The SMA architecture responds to Boeing enterprise needs to better secure a de-perimeterised network environment, as well as to Federal Aviation Administration (FAA) requirements that every step in the process of building an airplane be documented with the individual and equipment used [64]. HIP underpins the SMA architecture, and the identifiers in use in the network include machine (tool) host identifiers as well as temporary host identifiers linked to

[20]One of the authors has knowledge of this fact directly. However, at the time of this writing no contracts have been written and the prospective organisations do not want their identity to be revealed.

employee smart badges that are presented as part of the network log-in process.

Boeing has deployed an SMA pilot in its Everett, WA manufacturing facility. The architecture allows for network-based policy enforcement using Endboxes that can limit network connectivity using cryptographic identity rather than IP or MAC addresses. Complex trust relationships between contractors, employees, tools, and the enterprise network can be reflected in the network policy. One pilot deployment has been to secure the wireless network between the Boeing's 777 "crawlers" and their "controllers" (part of the implementation of the moving assembly line for the 777 aircraft) [63].

At the IETF, in the Peer-to-Peer SIP (P2PSIP) working group there are proposals by two independent organisations (Avaya and HIIT) to use HIP as a connectivity and security platform [65] [66]. The basic idea in these approaches is to apply HIP's ability to separate control and data traffic into different planes, using a distributed, peer-to-peer rendezvous service to establish HIP associations and to carry SIP packets, while running the actual data traffic directly over IP networks. The attractive features of HIP here seem to be opportunistic and leap-of-faith [57] security, ability to work through NATs, seamless support for mobility and multi-homing, and the ability to pass control traffic through an overlay-like rendezvous structure.

B. Standardisation situation

The HIP architecture document was published as RFC 4423 [1] in 2006. All the base protocol documents were published in 2008. The documents define the base exchange [28], using ESP transport format with HIP [29], the protocols and details for end-host mobility and multi-homing with HIP [46], the registration extension used in announcing and requesting HIP-based services [49], the rendezvous extension needed to use rendezvous services [50], HIP Domain Name System (DNS) extensions [44], and the details of using the HIP with legacy applications [32]. Additionally, a HIP Research Group at the Internet Research Task Force (IRTF) published a document specifying the issues related to NAT and firewall traversal [67].

After publication of the base RFCs, the HIP Working Group has been re-charted to focus on details for NAT traversal [34], native API [68], HIP-based overlay networks (HIP-BONE) [69], and carrying certificates in the HIP base exchange [70]. Several research drafts, such as the use of distributed hash tables (DHTs) for HIT-based lookups and a HIP experiment report [71] are being progressed at the HIP Research Group at IRTF.

The discussions of the HIP specifications at the IESG have increased the interest of individual IESG members to consider advancing HIP from its current experimental track to the standards track. Discussions on advancing HIP to Proposed Standard status commenced following the 74th IETF meeting (March 2009).

VIII. Conclusions

In this paper, we have discussed the Host Identity Protocol and architecture, and shown how it can be used to provide agile mobility and multi-homing over IPv4 and IPv6 networks, and how it can provide for enhanced privacy, traffic attribution, and protection from certain forms of unwanted traffic.

From an architectural point of view, the HIP architecture has been designed to restore the classical inter-networking invariants, allowing hosts to interconnect in the current immensely complex communication environment with IPv4, IPv6, NATs, and other middle boxes. HIP provides built-in, architected support for mobility, multi-homing (including multi-access), and baseline security. It enhances the IP architecture by introducing a Host Identity (HI) name space roughly between the IP layer and the transport protocols.

Beside the basic advantage of integrated mobility, multi-homing, and security support, HIP provides for a number of potential architectural extensions. The inherent delegation capability can be used to implement subnetwork-level mobility and multi-homing, as well as delegable application names. The architecture allows control traffic to be easily separated from data traffic, providing for enhanced protection against unwanted traffic.

At the time of this writing (Spring 2009), the HIP experimental IETF RFCs are being considered for promotion to standards-track. The protocol is used daily by a number of researchers and other early adopters. A few major governmental organisations in Europe are seriously considering the suitability of HIP for their internal use. The next step for HIP is to gain enough experimental experience with the approach and to evaluate whether its actual deployment will stand up to the architectural promise that it provides.

Acknowledgements

We want to thank Börje Ohlman and Jan Höller for their constructive comments on an early version of this paper. Mikko Särelä kindly helped to proofread an early version of the paper. Christian Vogt, Tobias Heer, and Martti Mäntylä provided numerous good comments that allowed us to improve the presentation. Peter Schoo provided encouragement and support throughout the whole writing process, without which this article would not have taken place.

References

[1] R. Moskowitz and P. Nikander, "Host Identity Protocol architecture," IETF RFC 4423, May 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4423.txt

[2] A. Gurtov, *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley and Sons, 2008.

[3] C. Perkins, "IP mobility support for IPv4," IETF, RFC 3334, Aug. 2002. [Online]. Available: http://tools.ietf.org/html/rfc3344

[4] C. Perkins, P. Calhoun, and J. Bharatia, "Mobile IPv4 challenge/response extensions (revised)," IETF RFC 4721, Jan. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4721.txt

[5] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[6] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols." Oct. 2007, work in progress, Expires in May, 2008. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-19.txt

[7] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," IETF RFC 4380, Feb. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4380.txt

[8] J. Chiappa, "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture." 1999. [Online]. Available: http://ana.lcs.mit.edu/ jnc/tech/endpoints.txt

[9] C. De Launois and M. Bagnulo, "The paths toward IPv6 multihoming," *IEEE Commun. Surveys Tutorials*, vol. 8, no. 2, pp. 38–51, 2006.

[10] D. Le, X. Fu, and D. Hogrefe, "A review of mobility support paradigms for the Internet," *IEEE Commun. Surveys Tutorials*, vol. 8, no. 2, pp. 38–51, 2006.

[11] D. Clark, "Application design and the end-to-end arguments," May 2007. [Online]. Available: http://google.com

[12] B. Carpenter, "Internet transparency," IETF RFC 2775, Feb. 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2775.txt

[13] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) terminology and considerations," IETF RFC 2663, Aug. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2663.txt

[14] G. Tsirtsis and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)," IETF RFC 2766, Feb. 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2766.txt

[15] C. Aoun and E. Davies, "Reasons to move the Network Address Translator - Protocol Translator (NAT-PT) to historic status," IETF RFC 4966, Jul. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4966.txt

[16] Wikipedia, "Naming collision," Oct. 2007. [Online]. Available: http://en.wikipedia.org/wiki/Naming_collision

[17] P. Sathyanathan, "Interprocedural dataflow analysis - alias analysis," Ph.D. dissertation, Stanford University, Computer Systems Laboratory, Jun. 2001.

[18] T. Harrison, C. Williams, W. Mackrell, and R. Bunt, "Mobile multicast (MoM) protocol: multicast support for mobile hosts," in *Proc. Third Annual ACM/IEEE International Conference on Computing and Networking (MOBICOM'97)*, 1997, pp. 151–160.

[19] Z. Kovacshazi and R. Vida, "Host identity specific multicast," in *Proc. of the International Conference on Networking and Services (ICNS'07)*, 2007, pp. 1–9.

[20] J. Day, "Patterns in network architecture: A return to fundamentals," Mar. 2008.

[21] B. LaMacchia, "Security Attacks and Defences," Jan. 2005, presentation at 10th Meeting of IFIP WG. [Online]. Available: http://www.laas.fr/IFIPWG/Workshop&Meetings/47/WS/08-LaMacchia.pdf

[22] B. Ramsdell, "Secure/Multipurpose Internet Mail Extensions (S/MIME) version 3.1 message specification," IETF RFC 3851, Jul. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3851.txt

[23] D. Solove, *The Digital Person: Technology and Privacy in the Information Age*. New York University Press, Feb. 2004.

[24] A. Odlyzko, "Privacy, economics, and price discrimination on the internet," in *Proc. Fifth International Conference on Electronic Commerce*. New York:ACM Press, 2003, pp. 355–366. [Online]. Available: htpp;//www.dtc.umn.edu/ odlyzko/doc/eworld.html

[25] J. Ylitalo and P. Nikander, "BLIND: A complete identity protection framework for end-points," in *Proc. Twelfth International Workshop on Security Protocols*, Apr. 2004.

[26] L. Takkinen, "Host Identity Protocol privacy management," Master's thesis, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, 2006.

[27] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," IETF RFC 2782, Feb. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2782.txt

[28] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Experimental Host Identity Protocol (HIP)," IETF RFC 5201, Apr. 2008.

[29] P. Jokela, R. Moskowitz, and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)," IETF RFC 5202, Mar. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5202

[30] H. Tschofenig, M. Shanmugam, and F. Muenz, "Using SRTP transport format with HIP: draft-tschofenig-hiprg-hip-srtp-02," Mar. 2006, work in progress. Expires in September, 2006.

[31] M. Baugher, D. A. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The secure real-time transport protocol (SRTP)," IETF, RFC 3711, Mar. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3711.txt

[32] T. R. Henderson, P. Nikander, and M. Komu, "Using the Host Identity Protocol with legacy applications," IETF RFC 5338, Sep. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5338.txt

[33] P. Nikander, J. Laganier, and F. Dupont, "An IPv6 prefix for overlay routable cryptographic hash identifiers (ORCHID)," IETF, RFC 4843, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4843.txt

[34] M. Komu, T. Henderson, H. Tschofenig, J. Melen, and A. Keranen, "Basic HIP Extensions for Traversal of Network Address Translators," Mar. 2009, work in progress.

[35] T. Kivinen, B. Swander, A. Huttunen, and V. Volpe, "Negotiation of NAT-traversal in the IKE," IETF RFC 3947, Jan. 2005. [Online]. Available: http://tools.ietf.org/html/rfc3947.txt

[36] A. Huttunen, B. Swander, V. Volpe, L. DiBurro, and M. Stenberg, "UDP encapsulation of IPsec ESP packets," Internet Engineering Task Force, RFC 3948, Jan. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3948.txt

[37] H. Tschofenig, M. Shanmugam, and M. Stiemerling, "Traversing HIP-aware NATs and firewalls: Problem statement and requirements: draft-tschofenig-hiprg-hip-natfw-traversal-06," Jul. 2007, work in progress. Expires in January, 2008.

[38] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proc. IEEE Symposium on Security and Privacy*, 1997.

[39] W. Eddy, "TCP SYN flooding attacks and common mitigations," Internet Engineering Task Force, RFC 4987, Aug. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4987.txt

[40] P. Nikander, J. Arkko, and B. Ohlman, "Host identity indirection infrastructure (Hi3)," in *Proc. Second Swedish National Computer Networking Workshop 2004 (SNCNW2004)*, Karlstad, Sweden, Nov. 2004.

[41] T. Aura, A. Nagarajan, and A. Gurtov, "Analysis of the HIP base exchange protocol," in *Proc. Tenth Australasian Conference in Information Security and Privacy. Brisbane, Australia*. Springer, Jul. 2005, pp. 481–493.

[42] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE-protocols." in *CRYPTO*, Santa Barbara, California, USA, Aug. 2003, pp. 400–425.

[43] P. Nikander and J. Melen, "A bound end-to-end tunnel (BEET) mode for ESP: draft-nikander-esp-beet-mode-09," Aug. 2008, work in progress. [Online]. Available: http://tools.ietf.org/html/draft-nikander-esp-beet-mode-09

[44] P. Nikander and J. Laganier, "Host Identity Protocol (HIP) domain name system (DNS) extension," IETF RFC 5205, Mar. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5205

[45] T. Heer, "LHIP: Lightweight Authentication for the Host Identity Protocol (HIP)," Master's thesis, University of Tubingen, Protocol-Engineering&Distributed Systems research group, 2006.

[46] P. Nikander, T. Henderson, C. Vogt, and J. Arkko, "End-host mobility and multihoming with the Host Identity Protocol (HIP)," IETF RFC 5206, Apr. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5206

[47] A. Gurtov, M. Komu, and R. Moskowitz, "Host Identity Protocol (HIP): Identifier/locator split for host mobility and multihoming," *Internet Protocol Journal*, vol. 12, no. 1, pp. 27–32, Mar. 2009.

[48] P. Nikander, J. Arkko, T. Aura, G. Montenegro, and E. Nordmark, "Mobile IP version 6 route optimization security design background," IETF RFC 4225, Dec. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4225.txt

[49] J. Laganier, T. Koponen, and L. Eggert, "Host Identity Protocol (HIP) registration extension," IETF RFC 5203, Apr. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5203

[50] J. Laganier and L. Eggert, "Host Identity Protocol (HIP) rendezvous extension," IETF RFC 5204, Mar. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5204

[51] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session traversal utilities for (NAT) (STUN)," Jul. 2008, work in progress. [Online]. Available: draft-ietf-behave-rfc3489bis-18.txt

[52] J. Rosenberg, "Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN)," Jul. 2007, work in progress. [Online]. Available: draft-ietf-behave-turn-04.txt

[53] H. Tschofenig and D. Wing, "Utilizing Interactive Connectivity Establishment (ICE) for the Host Identity Protocol (HIP): draft-tschofenig-hip-ice-00," Jun. 2007, work in progress. Expires in December, 2007.

[54] J. Ylitalo, P. Salmela, and H. Tschofeing, "SPINAT: Integrating IPsec into overlay routing," in *Proc. 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, Sep. 2005, pp. 315–326.

[55] P. Nikander, "An architecture for authorization and delegation in distributed objec-oriented agent systems," Ph.D. dissertation, Helsinki University of Technology, Jun. 1999.

[56] T. Koponen, A. Gurtov, and P. Nikander, "Application mobility with Host Identity Protocol," in *Proc. of NDSS Wireless and Security Workshop*. San Diego, CA, USA: Internet Society, Feb. 2005.

[57] J. Arkko and P. Nikander, "How to authenticate unknown principals without trusted parties," in *Proc. 10th International Workshop. Security Protocols. Cambridge, UK*. Springer, Apr. 2002, pp. 5–16.

[58] A. Gurtov, D. Korzun, A. Lukyanenko, and P. Nikander, "Hi3: An efficient and secure networking architecture for mobile hosts," *Computer Communications*, vol. 31, no. 10, pp. 2457–2467, Jun. 2008.

[59] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: secure overlay services," in *SIGCOMM*, 2002, pp. 61–72.

[60] P. Wang, P. Ning, and D. S. Reeves, "A k-anonymous communication protocol for overlay networks," in *Proc. 2nd ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS'07), Singapore*, Mar. 2007, pp. 45–56.

[61] A. Khurri, E. Vorobyeva, and A. Gurtov, "Performance of Host Identity Protocol on lightweight hardware," in *Proc. 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'07)*. New York, NY, USA: ACM, Aug. 2007.

[62] B. Estrem, "Secure mobile architecture (sma) vision & architecture," Feb. 2004, technical Study E041. [Online]. Available: http://www.opengroup.org/products/publications/catalog/e041.htm

[63] R. Paine, "Secure mobile architecture (SMA) for automation security," Jul. 2007. [Online]. Available: http://google.com

[64] "Boeing IT architect pushes Secure Mobile Architecture," Apr. 2006. [Online]. Available: http://www.networkworld.com/news/2006/050106-boeing-side.html

[65] E. Cooper, A. Johnston, and P. Matthews, "A distributed transport function in P2PSIP using HIP for multi-hop overlay routing: draft-matthews-p2psip-hip-hop-00," Jun. 2007, work in progress. Expired in December, 2007.

[66] J. Hautakorpi, G. Camarillo, and J. Koskela, "Utilizing HIP (Host Identity Protocol) for P2PSIP (Peer-to-peer Session Initiation Protocol): draft-hautakorpi-p2psip-with-hip-01.txt," Nov. 2007, work in progress.

[67] M. Stiemerling, J. Quittek, and L. Eggert, "NAT and firewall traversal issues of Host Identity Protocol (HIP) communication," IETF RFC 5207, Apr. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5207

[68] M. Komu and T. Henderson, "Basic Socket Interface Extensions for Host Identity Protocol (HIP), draft-ietf-hip-native-api-05.txt," Jul. 2008.

[69] G. Camarillo, P. Nikander, J. Hautakorpi, and A. Johnston, "HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment, draft-ietf-hip-bone-01.txt," Mar. 2009, work in progress.

[70] T. Heer and S. Varjonen, "HIP Certificates: draft-ietf-hip-cert-00," Oct. 2008, work in progress.

[71] T. R. Henderson and A. Gurtov, "HIP experiment report: draft-irtf-hip-experiment-05.txt," Mar. 2009.

Pekka Nikander received his M.Sc. and Ph.D (with distinction) in Computer Science from Helsinki University of Technology, Finland, in 1992 and 1999. At the present, he is Chief Scientist at Ericsson Research Nomadic Lab, an industrial research laboratory located in Jorvas, Finland. From 1999 to 2007 he was active in Internet standardisation at the IETF, chairing a few working groups and authoring some RFCs. He also served for a while at the Internet Architecture Board (IAB). Most recently, he has focused more on research on the future of the Internet architecture, with special attention to economic, end-user centric, and privacy aspects. He is a co-author of over 70 peer reviewed papers, 10 RFCs, and some 15 patents.

Andrei Gurtov received his M.Sc. and Ph.D. degrees in Computer Science from the University of Helsinki, Finland in 2000 and 2004. At the present, he is Principal Scientist leading the Networking Research group at the Helsinki Institute for Information Technology focusing on the Host Identity Protocol and next generation Internet architecture. He is co-chairing the IRTF research group on HIP and teaches as an adjunct professor at the Helsinki University of Technology. Previously, his research focused on the performance of transport protocols in heterogeneous wireless networks. In 2000-2004, he served as a senior researcher at TeliaSonera Finland contributing to performance optimization of GPRS/UMTS networks, intersystem mobility, and IETF standardization. In 2003, he spent six months as a visiting researcher in the International Computer Science Institute at Berkeley working with Dr. Sally Floyd on simulation models of transport protocols in wireless networks. In 2004, he was a consultant at the Ericsson NomadicLab. Dr. Gurtov is a co-author of over 80 publications including a book, research papers, patents, and IETF RFCs.

Thomas R. Henderson is a network researcher and Boeing Associate Technical Fellow in Boeing's Research & Technology division. He also is an Affiliate Professor in the Electrical Engineering department at the University of Washington. His research interests are currently focused on protocols and software for wireless-based Internets, and network simulation. Presently, he is the Principal Investigator (PI) of an applied research program with the U.S. Office of Naval Research (ONR), and the PI of the NSF-funded ns-3 simulation project. For the past five years he has served as co-chair of the Internet Research Task Force (IRTF) HIP research group. Tom holds a Ph.D. in EECS from the University of California, Berkeley, and an MSEE and BSEE from Stanford University.